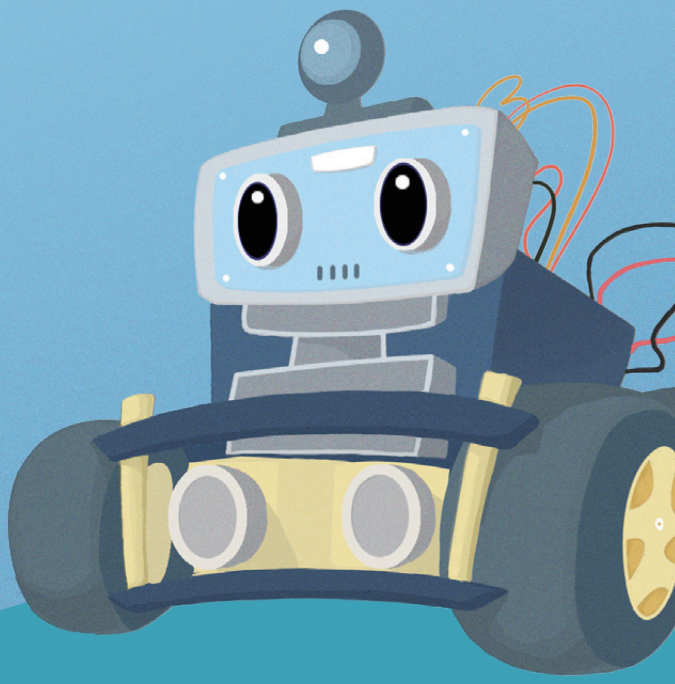
ELEGOO

5

SMART ROBOT
CAR V4.0 WITH CAMERA



Obstacle-avoidance Mode

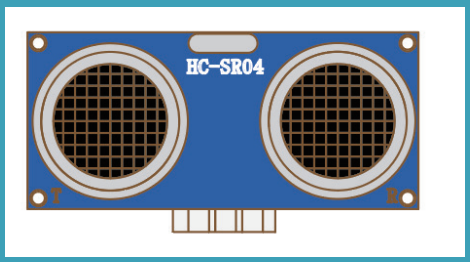
Introduction:

In this lesson, we will teach you how to achieve the Obstacle-avoidance Mode of the Smart Robot Car and make it change the moving direction when encountering obstacles.

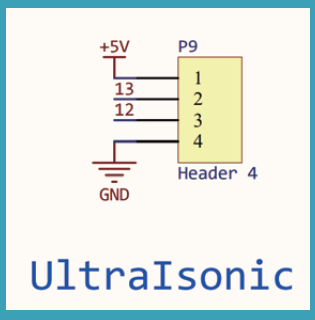
Preparation:

A Smart Robot Car (with battery)
A USB Wire

In our kit, the Obstacle-avoidance Mode is implemented with the ultrasonic module. The principle of ultrasonic distance measurement is that the transmitter emits ultrasonic waves in a certain direction and starts timing at the same time of the launch moment. The ultrasonic waves travel through the air and returns immediately when encountering obstacles. Then the ultrasonic receiver stops timing immediately when receiving the reflected waves. According to the recorded time and the known condition that ultrasonic propagation speed in the air is 340m/s, the distance between the ultrasonic module and the obstacle can be obtained.



Please open the last folder for details: Related chip information -> SmartRobot-Shield



It can be seen from the figure that the ultrasonic module is connected to D12 and D13 on the UNO board.

Then please open Demo1:

Next, let's take a look at the definition of the relative pins and the variables of ultrasonic.

C:/ // in DeviceDriverSet_xxx0.h

```
class DeviceDriverSet_ULTRASONIC
{
public:
 void DeviceDriverSet_ULTRASONIC_Init(void);
 void DeviceDriverSet_ULTRASONIC_Test(void);
 void DeviceDriverSet_ULTRASONIC_Get
 (uint16_t *ULTRASONIC_Get /*out*/);

private:
#define TRIG_PIN 13
// Arduino pin tied to trigger pin on the ultrasonic sensor.
#define ECHO_PIN 12
// Arduino pin tied to echo pin on the ultrasonic sensor.
#define MAX_DISTANCE 200
// Maximum distance we want to ping for (in centimeters).
//Maximum sensor distance is rated at 400-500cm.
};
```

Then, initial the pins of ultrasonic.

C:/ // in DeviceDriverSet_xxx0.cpp

```
void DeviceDriverSet_ULTRASONIC::DeviceDriverSet_ULTRASONIC_Init(void)
{
 pinMode(ECHO_PIN, INPUT); //Ultrasonic module initialization
 pinMode(TRIG_PIN, OUTPUT);
}
```

And then call it in the Setup() function:

C:/ // in Demo1.ino

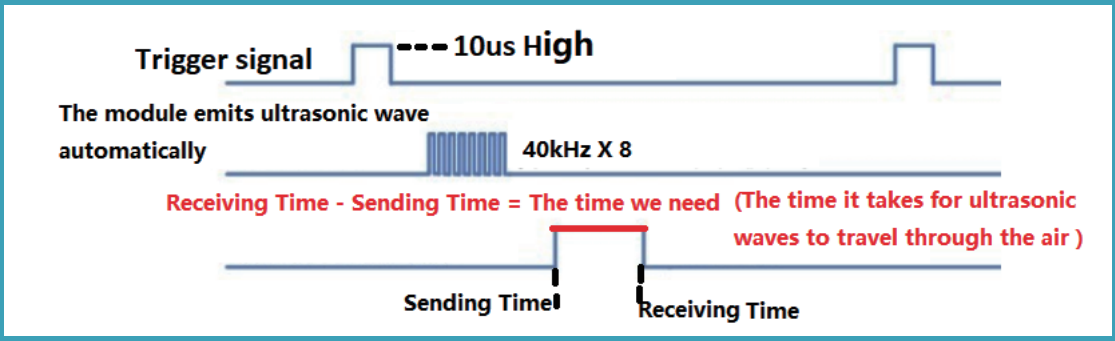
```
void setup()
{
 myUltrasonic.DeviceDriverSet_ULTRASONIC_Init();
}
```

✚ Next, let’ s look at the process of acquiring the measurement data through ultrasonic module.

1、Use Arduino to employ digital pin to give at least 10μs high level signal to Trig pin of SR04 to trigger the distance measurement function of SR04 module.

2、After triggering, the module will automatically send eight 40KHz ultrasonic pulses and detect whether there is a signal return. This step is done internally by the module automatically.

3、If a signal returns, the Echo pin will output high level, and the duration of the high level is the time from launch to return of the ultrasound. At this point, we can use the pulseIn() function to obtain the distance measurement data, and calculate the actual distance from the object being measured.



✚ We will use the pulseIn(pin,value) function during the programming. pulseIn(pin, value): The pulse width used to detect the high and low level output of the pin, i.e., the time we need.

Pin: The pin used to obtain the pulse.

Value: Type of pulse, HIGH or LOW

✚ Since the default unit of the pulseIn() function is “us”, and the propagation velocity of ultrasonic waves in the air is about 340m/s, unit conversions is needed.

Because,
the distance traveled by the ultrasonic wave = the distance from the launch point to the end + the distance from the end back to the launch point.
Therefore,
the distance traveled by the ultrasonic wave = pulseIn()/29.15/2 ≈ pulseIn()/58

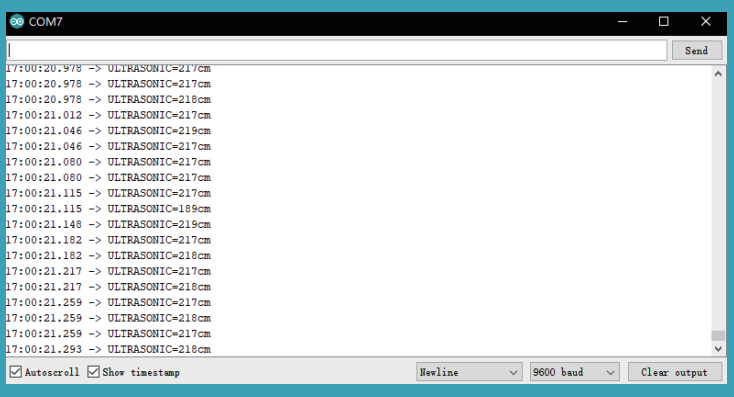
C:/ // in DeviceDriverSet_xxx0.cpp

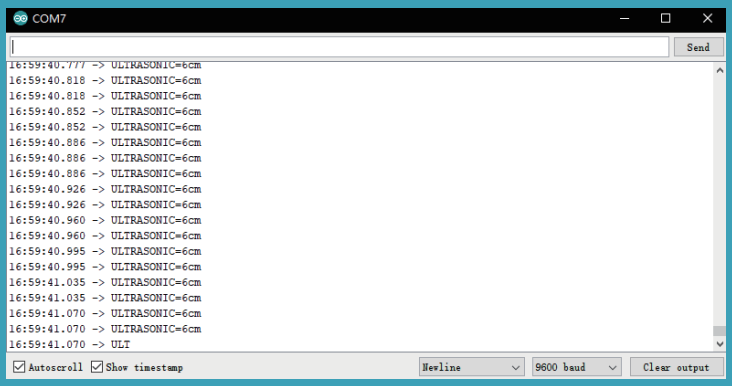
```
void DeviceDriverSet_ULTRASONIC::DeviceDriverSet_ULTRASONIC_Test(void)
{

 unsigned int tempda = 0;
 digitalWrite(TRIG_PIN, LOW);
 delayMicroseconds(2);
 digitalWrite(TRIG_PIN, HIGH);
 delayMicroseconds(10);
 digitalWrite(TRIG_PIN, LOW);
 tempda = ((unsigned int)pulseIn(ECHO_PIN, HIGH) / 58);

 Serial.print("ULTRASONIC=");
 Serial.print(tempda); // Convert ping time to distance and print result
                       //(0 = outside set distance range, no ping echo)
 Serial.println("cm");
}
```

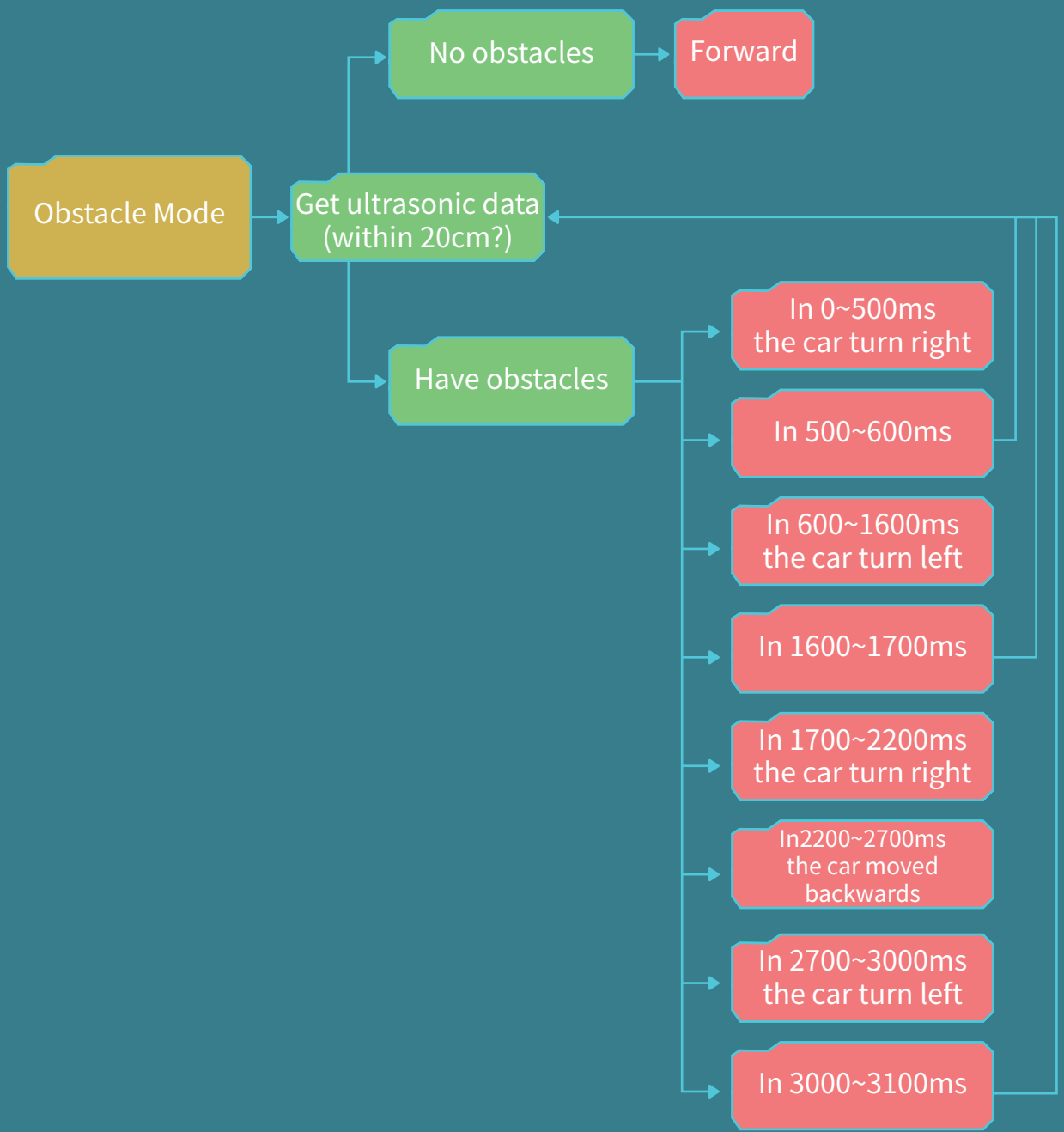
✚ Upload program. (Please toggle the “Upload-Cam” button to “Upload” when uploading the program.) Please open the serial port after the program has been uploaded successfully. If there is no obstacle in front, the ultrasonic distance will be shown as more then 200. And if you try to put your hand in front of the ultrasonic module, it will show the distance between your hand and the ultrasonic module.





✚ At this point, the ultrasonic module driver is completed. Now, let's take a look at the overall framework of the implementation principle of the Obstacle-avoidance Mode using the ultrasonic module and then analyze the program:

Implementation principle of Obstacle



✚ Please open Demo2 in the current folder:

✚ Let’ s look at the definition of the relative functions and variables of Obstacle-avoidance Mode:

C:/
```
// in ApplicationFunctionSet_xxx0.h
#ifndef _ApplicationFunctionSet_xxx0_H_
#define _ApplicationFunctionSet_xxx0_H_

#include <arduino.h>

class ApplicationFunctionSet
{
public:
  void ApplicationFunctionSet_Init(void);
  void ApplicationFunctionSet_Obstacle(void);

private:
  volatile uint16_t UltrasoundData_mm;
  volatile uint16_t UltrasoundData_cm;
  boolean UltrasoundDetectionStatus = false;
public:
  boolean Car_LeaveTheGround = true;
  const int ObstacleDetection = 20;
};
extern ApplicationFunctionSet Application_FunctionSet;
#endif
```

✚ Then, as for writing the program of Obstacle-avoidance Mode, the situation described on the flow chart when encountering obstacles can be listed down one by one.

C:/
```
// in ApplicationFunctionSet_xxx0.h
 void ApplicationFunctionSet::ApplicationFunctionSet_Obstacle(void)
{.....}
```

✚ Upload program. (Please toggle the “Upload-Cam” button to “Upload” when uploading the program.) After the program has been uploaded successfully, please place the car on the ground steadily and then turn on the switch. The Smart Robot Car will moving forward by default if there is no obstacle in front. And it will implement the Obstacle-avoidance Mode intelligently depends on different situations when encountering the obstacle.